

# OCI Exports metrics to Prometheus Solution

---

[Jingsong.liu@oracle.com](mailto:Jingsong.liu@oracle.com)

2022 年 4 月 18, Version 1.0

Copyright © 2023, Oracle and/or its affiliates

Public

## Table of contents

Table of contents	2
产品版本	3
Overview	4
Field explanation	5
How to get namespace and MQL	5
How to get compartmentId	7
Oci_exporter Metric type	8
Prometheus config	8
Run OCI exporter	8

## 产品版本

OCI 云平台

prometheus-2.43.0.windows-amd64

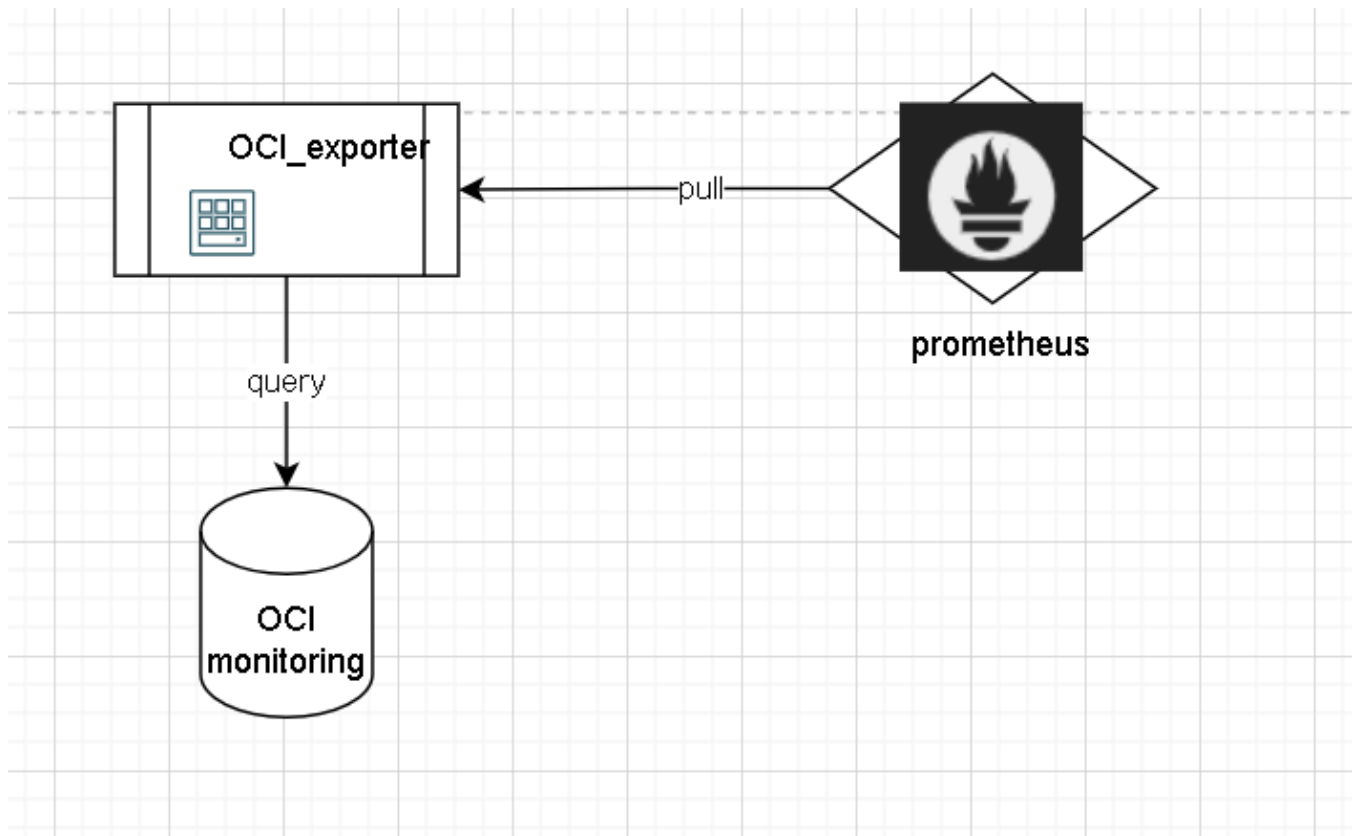
Golang 1.20, go.mod as below

```
github.com/oracle/oci-go-sdk/v65 v65.36.0  
github.com/prometheus/client_golang v1.15.0
```

## Overview

Prometheus is a very popular monitoring solution in industry, so we need to have the requirement that exports metrics to Prometheus.

About the usage we only need to Get the binary and make metrics.yaml file Prepared and then run the application In a virtual machine. Let the Prometheus discover These instances of exporters. then the metrics will be exported to Prometheus



The binary of windows and linux can be downloaded from github

[https://github.com/munger1985/OCI-Auto-Scripts/tree/main/oci\\_exporter](https://github.com/munger1985/OCI-Auto-Scripts/tree/main/oci_exporter)

For usage, we can simply add metrics in metrics.yaml

```
---
metrics:
- name: oci_all_lb_bytesent
  help: "The number of bytes sent from all load balancer."
  type: many
  interval: 3
  namespace: oci_lbaas
  mql: BytesSent[1m].mean()
  compartmentId: ocid1.compartment.oc1..aaaaaaaahr7aicqtodxmcf0r6pbqn3hvsngpftozyxqzw36gj4kh3w3kkj4q
  label1: dev
  label2: high
- name: oci_vm1_mem
```

```

help: "vm1 mem usage."
type: single
interval: 1
namespace: oci_computeagent
mql: MemoryUtilization[1m]{resourceId = "ocid1.instance.oc1.ap-singapore-
1.anzws1jrak7gbriccijo42npujwjrub3j6e6fcoe7ijtpcn4krhndgr3f2da"}.mean()
compartmentId: ocid1.compartment.oc1..aaaaaaaahr7aicqtodxmcf6pbqn3hvsngpftozyxqw36gj4kh3w3kkj4q
label1: dev
label2: high
# - name: oci_vm_mem
# help: "oci mem usgae ."
# type: single
# interval: 1
# namespace: oci_computeagent
# mql: DiskIopsWritten[1m]{resourceId = "ocid1.instance.oc1.ap-singapore-
1.anzws1jrak7gbriccijo42npujwjrub3j6e6fcoe7ijtpcn4krhndgr3f2da"}.rate()
# label:
# a: b

```

You can comment some metric whenever you want

## Field explanation

Name: name of the metric

Help: description

Type: if this metric will get you many datapoints, then it is many, otherwise single, many is used to query many resource in one go, you can all use many.

Interval: export query data in this interval, unit: minute

Namespace: for query data

compartmentId: compartment of the target resource

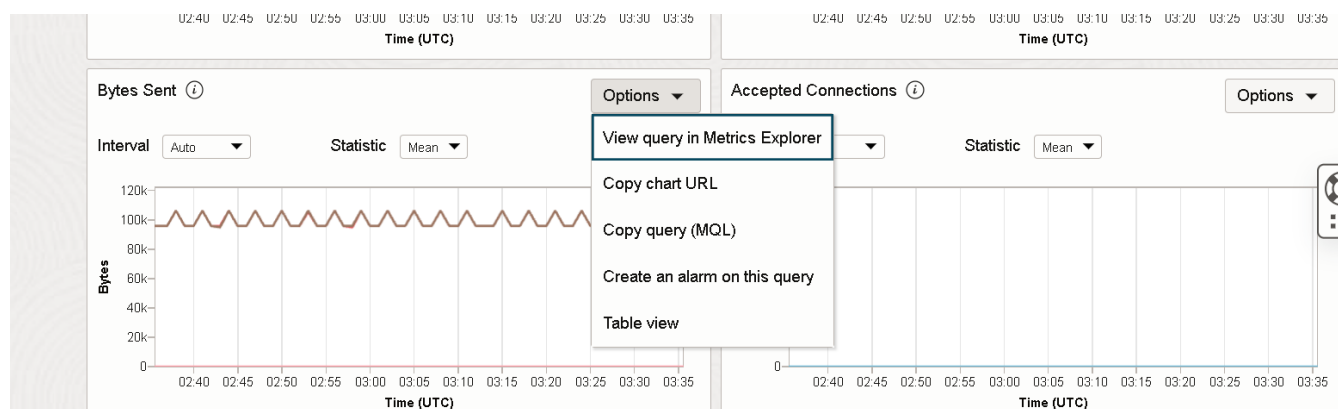
Label1: value of custom label, can be used for Prometheus filter

Label2: value of custom label, can be used for Prometheus filter

## How to get namespace and MQL

The screenshot shows the OCI console interface. On the left, the 'Resources' sidebar has the 'Metrics' tab selected, indicated by a red arrow. The main content area is titled '11 metrics'. At the top, there's a 'Logs' section with 'Error logs: Not enabled' and 'Access logs: Not enabled', both with information icons, and a link to 'Learn more about load balancer logging'. Below this, the 'Inbound Requests' metric is displayed with a graph. The graph shows a line chart with a tooltip that reads 'Number of incoming client requests to the load balancer'. The graph has controls for 'Interval' (set to 'Auto') and 'Statistic' (set to 'Mean'). To the right of the graph, there's a section for 'Active Connections' with similar controls. The top of the graph area includes 'Start time' (May 5, 2023 02:35:34 UTC), 'End time' (May 5, 2023 03:35:34 UTC), and 'Quick selects' (Last hour), along with a 'Reset' button.

## Select your target metric open explorer



## Copy MQL



Here is the namespace

**Query 1**

BytesSent[1m]{resourceId = "ocid1.loadbalance...}

Add Query

**Query 1**

Compartment: ChinaTech  
sehubjapacprod (root)/ChinaTech

Metric namespace: oci\_lbaas

Metric name: BytesSent

Interval: 1 minute

Statistic: Mean

Metric dimensions

Dimension name: resourceId

Dimension value: ocid1.loadbalancer.oc1.ap-singapore-1.aaaaaaaai2jz

☐ Aggregate metric streams

+ Addition

## How to get compartmentId

ORACLE Cloud chinattech Singapore (Singapore)

You are using a fixed-shape (dynamic) load balancer. Oracle will retire the ability to create new dynamic shape load balancers on Thu, 11 May 2023 00:00:00 UT. Oracle recommends using the cost-efficient flexible load balancers.

### Load balancers in ChinaTech Compartment

Load balancers provide automated traffic distribution from one entry point to multiple servers reachable from your virtual cloud network (VCN). They improve resource utilization, facilitate scaling, and help ensure high availability.

Create load balancer

Name	Type	State	IP address	Sh
<a href="#">36f8f4e8-2037-4b67-885b-df5cc9c7be60</a>	Load balancer	Active	138.2.87.224 (public)	10
<a href="#">6459df0c-70b3-4b00-a25f-e62fcf301f35</a>	Load balancer	Active	129.150.49.220 (public)	10
<a href="#">33f1e1ec-ec34-4e8f-a7b5-a56c098cd716</a>	Load balancer	Active	138.2.68.109 (public)	10
<a href="#">6c876f91-257a-4f5f-acc1-393e16b68d93</a>	Load balancer	Active	168.138.184.212 (public)	10
<a href="#">c9a5901a-9694-4d24-bdd3-7131b5ff5db2</a>	Load balancer	Active	158.178.237.157 (public)	10

Search the name in search bar

Compartment Information Tags

**Parent Compartment:** [sehubjapacprod \(root\)](#)

**OCID:** ...3kkj4q [Show](#) [Copy](#)

**Authorized:** Yes

**Created:** Fri, Aug 14, 2020, 06:57:20 UTC

After you get all these, fill them in the metrics.yaml

The binary will try to export those metrics to be pulled by Prometheus

## Oci\_exporter Metric type

I have designed 2 types.

Single: if this MQL can only get 1 datapoint for one resource

Many: will get us many data points for many resources

## Prometheus config

In order to let it discover metrics, we write config (prometheus.yaml) for it, **yellow** part is what we have added

```
global:
  scrape_interval:      15s # By default, scrape targets every 15 seconds.

  # Attach these labels to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    monitor: 'codelab-monitor'

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from
  # this config.
  - job_name: 'prometheus'

    # Override the global default and scrape targets from this job every 5 seconds.
    scrape_interval: 5s

    static_configs:
      - targets: ['localhost:9090']
  - job_name:      'oci-custom-job'

    # Override the global default and scrape targets from this job every 5 seconds.
    scrape_interval: 60s

    static_configs:
      - targets: [ 'localhost:8081' ]
        labels:
          group: 'production2'
```

## Run OCI exporter

We need to run application against API key so we need to prepare key first.



In your application directory, we have a binary executable and metrics.yaml file.



Oci\_exporter.exe will listen on port 8080

When we access [127.1.1.1:8080/metrics](http://127.1.1.1:8080/metrics)

We can get metrics defined in metrics.yaml

We can add parameter to run on other port

### Windows

oci\_exporter.exe -listen-address=:8081 -config c:\adasdsad\config

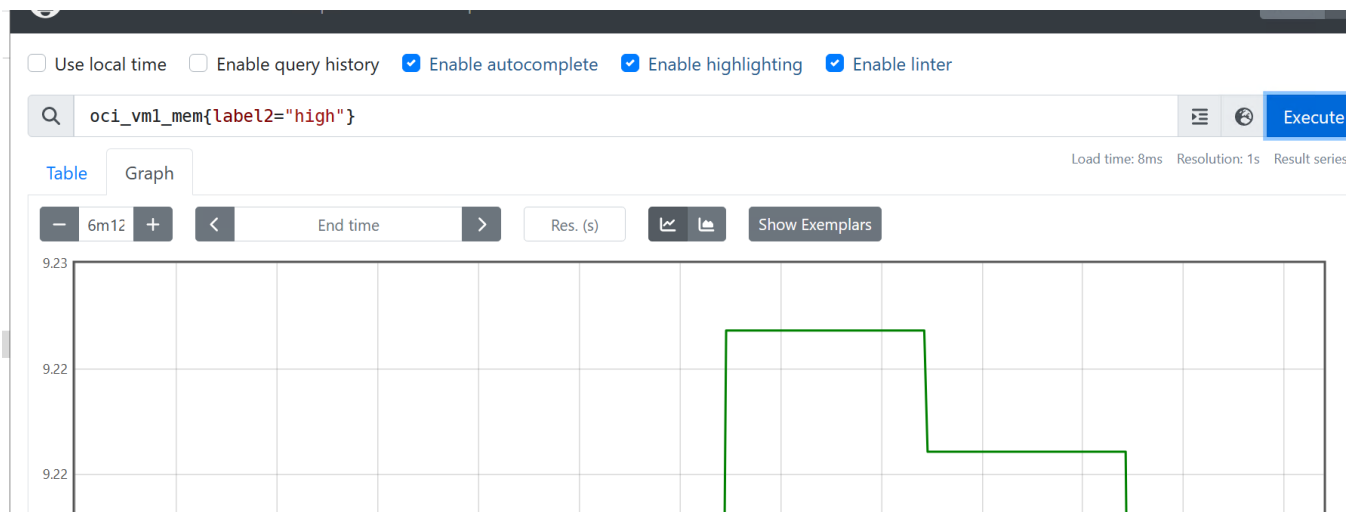
will listen on 8081

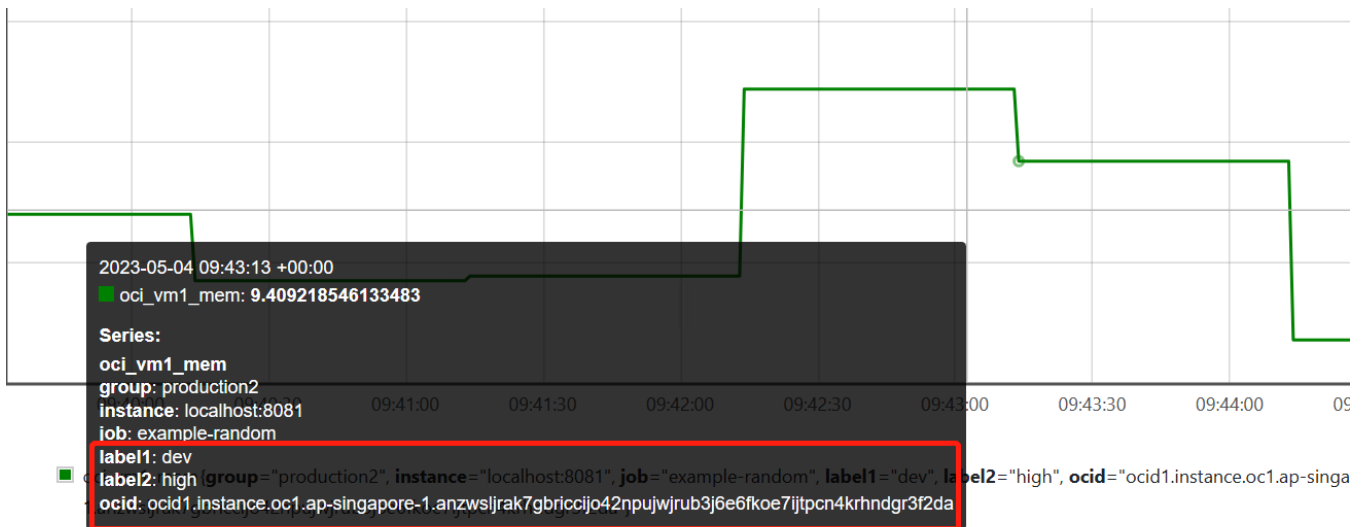
if oci\_exporter.exe -listen-address=0.0.0.0:8081 -config c:\adasdsad\config

will bind all ips of local to listen on 8081

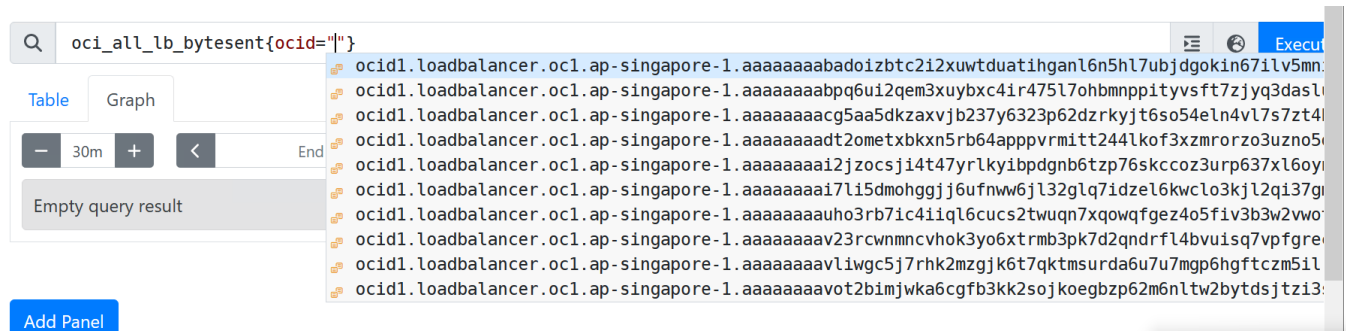
when we run Prometheus, we open dashboard

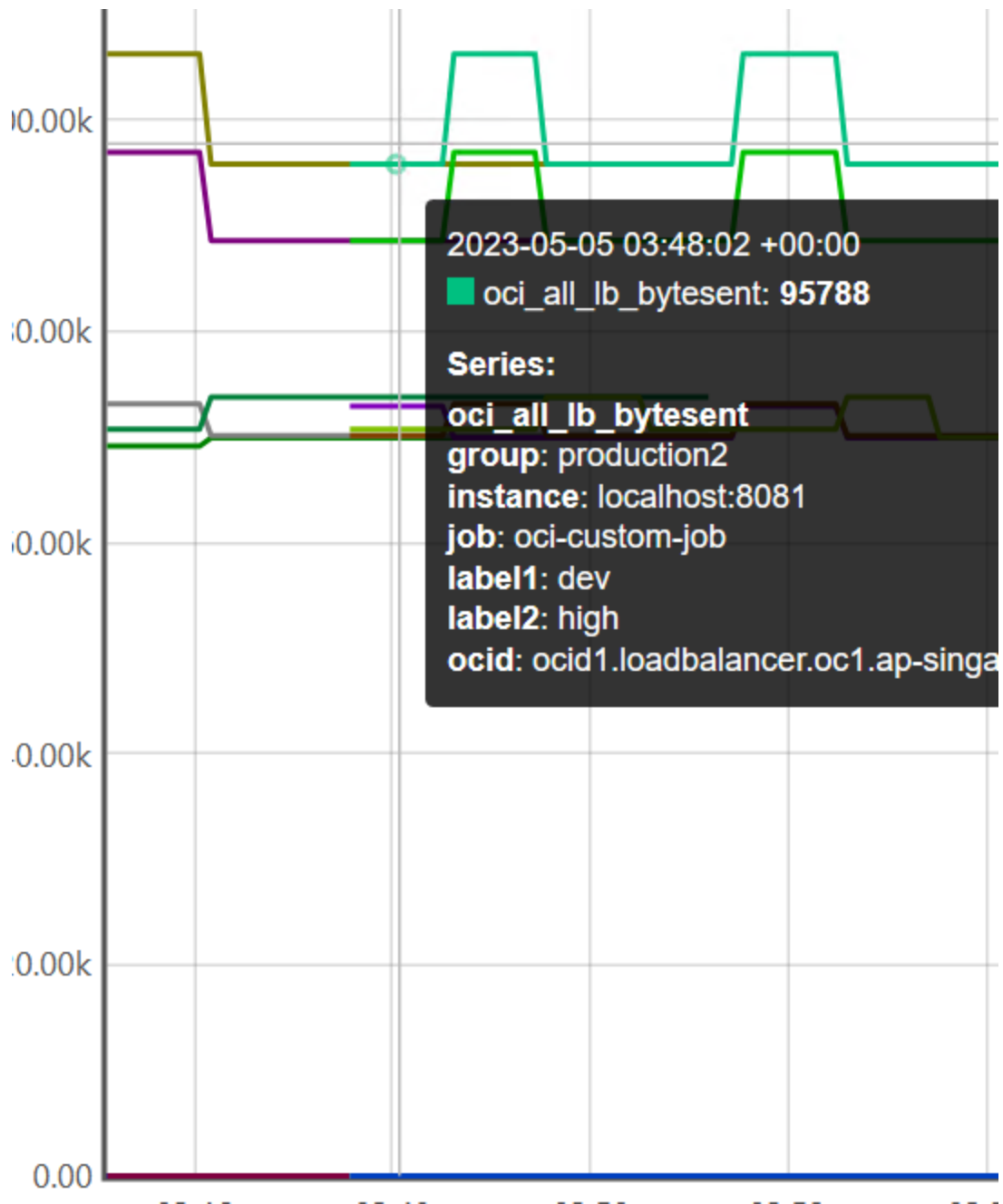
type the metric name and filter with the label





When we type promQL we get auto completions





- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaabadoizbtc2i2xuwtduatihganl6n5hl7ubjdgokin67ilv5mniya"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaabpq6ui2qem3xuybxc4ir475l7ohbmnpityvsft7zjq3dasluja"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaacg5aa5dkzaxvjb237y6323p62dzrkyjt6so54eln4vl7s7zt4bea"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaadt2ometxbkx5rb64apppvrmitt244lkof3xzmrorzo3uzno5dma"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaai2jzocsji4t47yrlkyibpdgnb6tzip76skccoz3urp637xl6oynya"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaai7li5dmohggjj6ufnww6jl32glq7idzel6kwclo3kjl2qi37gmaq"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaauho3rb7ic4iiql6cucs2twuqn7xqowqfgez4o5fiv3b3w2vwotla"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaav23rcwnmncvhok3yo6xtrmb3pk7d2qndrfl4bvuisq7vpfgrecgq"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaavliwgc5j7rhk2mzgjk6t7qktmsurda6u7u7mgp6hgftczm5ilrta"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaavot2bimjwka6cgfb3kk2sojkoegbzip62m6nlw2bytdsjtzi3saq"}
- oci\_all\_lb\_bytesent{group="production2", instance="localhost:8081", job="€1.aaaaaaaabadoizbtc2i2xuwtduatihganl6n5hl7ubjdgokin67ilv5mniya"}

## Linux

```
./oci_exporter -config=/home/opc/.oci/config -listen-address=:8081
```

## Docker

### Dockerfile

```
...
FROM golang

# Set environment variables
```

```
# Update the package repository and install packages
```

```
RUN apt-get update && \  
    apt-get install -y \  
    iproute2 \  
    wget \  
    git \  
    build-essential \  
    && apt-get clean \  
    && rm -rf /var/lib/apt/lists/*
```

```
# Set a working directory
```

```
WORKDIR /app
```

```
# Copy files into the container (if needed)
```

```
COPY . /app
```

```
RUN chmod +x /app/oci_exporter
```

```
````
```

## build

```
docker build -t oci_exporter .
```

## run

```
docker run  oci_exporter  /app/oci_exporter -config=/app/config
```

- /app/config is api key config file, need to mount volume into docker
- need to make sure key.pem in config file is existed.